# LAB 2

1. ***Your first report – a model***
2. **Android Studio – short presentation**
3. **Build a First Basic App –example**

**Phd Lecturer Liviu Octavian Mafteiu-Scai**

# *Your first report – a model*

Title – the name of the app

+ “First report – Proposed mobile app”

Student full name, study year

Course name

Submission date

Supervisor name

Size of your first report: about 5 - 10 pages with TNR 12

# *Your first report – a model*

## 1. Abstract (max. 100 words)

*! - report abstract, not a description of the problem/app*

## 2. Goal and users

## 3. Introduction

- *A short description of the problem*

## 4. State of art

- A review for several similar apps

## *Your first report – a model*

5. <u>What is the original contribution of the author?</u>

- enumeration and purpose

6. <u>Development plan</u>

- some diagrams about functionality and user user interface are enough *(see Software Engineering course)*
- SDKs and other tools that will be used

7. References

- Books and used links for all idea that you used for your

DEADLINE: BEFORE 4th LAB

# *Your first report – a model*

DEADLINE:

Before 4th lab

You must put the first report here: https://groups.google.com/g/programmingformobiledevices_ie2_spring2019/c/NgCzvCzmw7g as a reply to *First report's box* conversation, using *Attach* option

The report must be a doc/docx document, TNR12 about 5-10 pages

# What is Android Studio?

**Android Studio** is the official IDE for Android application development, based on *IntelliJ IDEA*

replace Eclipse Android Development Tools (ADT)
-first Google's IDE for native Android apps develop.

ver. 0.1 May 2013, ver. 0.8 released in June 2014.
The first stable version: 1.0 released in Dec. 2014
Curent: 1.4 sept 2015

available for Windows, Mac OS X and Linux.

a Java IDE developed by JetBrains
-known as IntelliJ;
-ver. 1 ian. 2001, 14.1 sept. 2015;
-other software based on it:
AppCode, PhpStorm, PyCharm, RubyMine, WebStorm, etc

| | Windows | OS X | Linux |
|---|---|---|---|
| OS version | Microsoft Windows 10/8.1/8/7/Vista (32 or 64 bit) (64 recommended) | Mac OS X 10.8.5 or higher, up to 10.10 to up 10.10.2 up 10.10.3 on 10.10.5 (Yosemite) | GNOME or KDE or Unity desktop on Ubuntu or Fedora or GNU/Linux Debian |
| RAM | 2 GB RAM minimum, 4 GB RAM recommended, 16 GB RAM good | | |
| Disk space | 500 MB disk space | | |
| Space for Android SDK | At least 1 GB for Android SDK, emulator system images, and caches | | |
| JDK version | Java Development Kit (JDK) 7 or higher 64 bit (full JDK not only JRE) | | |
| Processor | I5 orI7 Intel that support Intel® Hardware Accelerated Execution Manager (Intel® HAXM) | | |
| Screen resolution | 1280x800 minimum screen resolution | | |

# Downloading and Installing

- JDK (not JRE) must be installed before:

www.oracle.com/technetwork/java/javase/downloads/index.html

on Windows:

New to Java
Community
Java Magazine

- Java Developer Day hands-on workshops (free) and other events
- Java Magazine

JDK MD5 Checksum

Looking for JDK 8 on ARM?
JDK 8 for ARM downloads have moved to the JDK 8 for ARM download page.

**Java SE Development Kit 8u25**

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Accept License Agreement  Decline License Agreement

| Product / File Description | File Size | Download |
|---|---|---|
| Linux x86 | 135.24 MB | jdk-8u25-linux-i586.rpm |
| Linux x86 | 154.88 MB | jdk-8u25-linux-i586.tar.gz |
| Linux x64 | 135.6 MB | jdk-8u25-linux-x64.rpm |
| Linux x64 | 153.42 MB | jdk-8u25-linux-x64.tar.gz |
| Mac OS X x64 | 209.13 MB | jdk-8u25-macosx-x64.dmg |
| Solaris SPARC 64-bit (SVR4 package) | 137.01 MB | jdk-8u25-solaris-sparcv9.tar.Z |
| Solaris SPARC 64-bit | 97.14 MB | jdk-8u25-solaris-sparcv9.tar.gz |
| Solaris x64 (SVR4 package) | 137.11 MB | jdk-8u25-solaris-x64.tar.Z |
| Solaris x64 | 94.24 MB | jdk-8u25-solaris-x64.tar.gz |
| Windows x86 | 157.26 MB | jdk-8u25-windows-i586.exe |
| Windows x64 | 169.62 MB | jdk-8u25-windows-x64.exe |

**Java SE Development Kit 8u25 Demos and Samples Downloads**

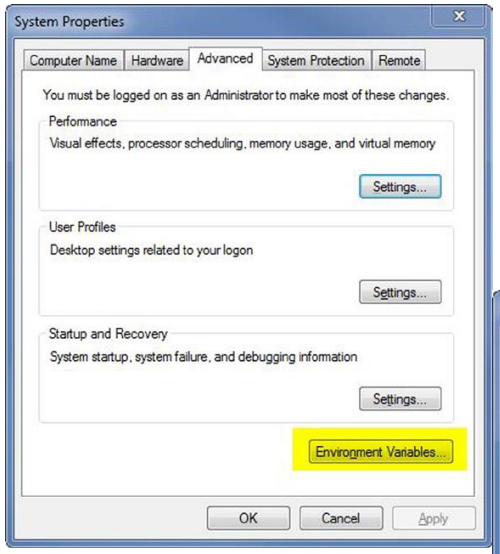
Java SE Development Kit 8u25 Demos and Samples Downloads are released under the Oracle BSD License.

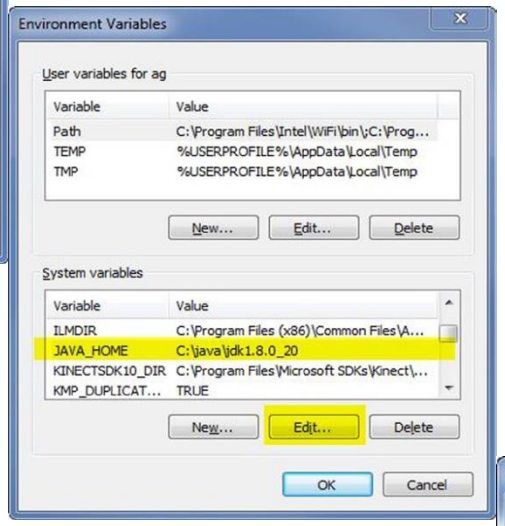
| Product / File Description | File Size | Download |
|---|---|---|
| Linux x86 | 58.63 MB | jdk-8u25-linux-i586-demos.rpm |
| Linux x86 | 58.52 MB | jdk-8u25-linux-i586-demos.tar.gz |

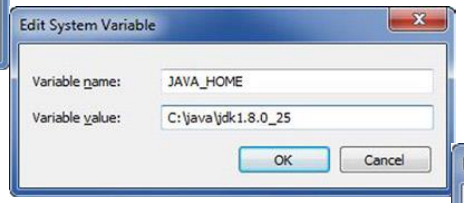
# Downloading and Installing

- JDK installation: navigate to the location where installation file was download and execute that file, preferably in C:\Program Files\Java\.

- after this, Environmental Variables on Windows must be configurated:

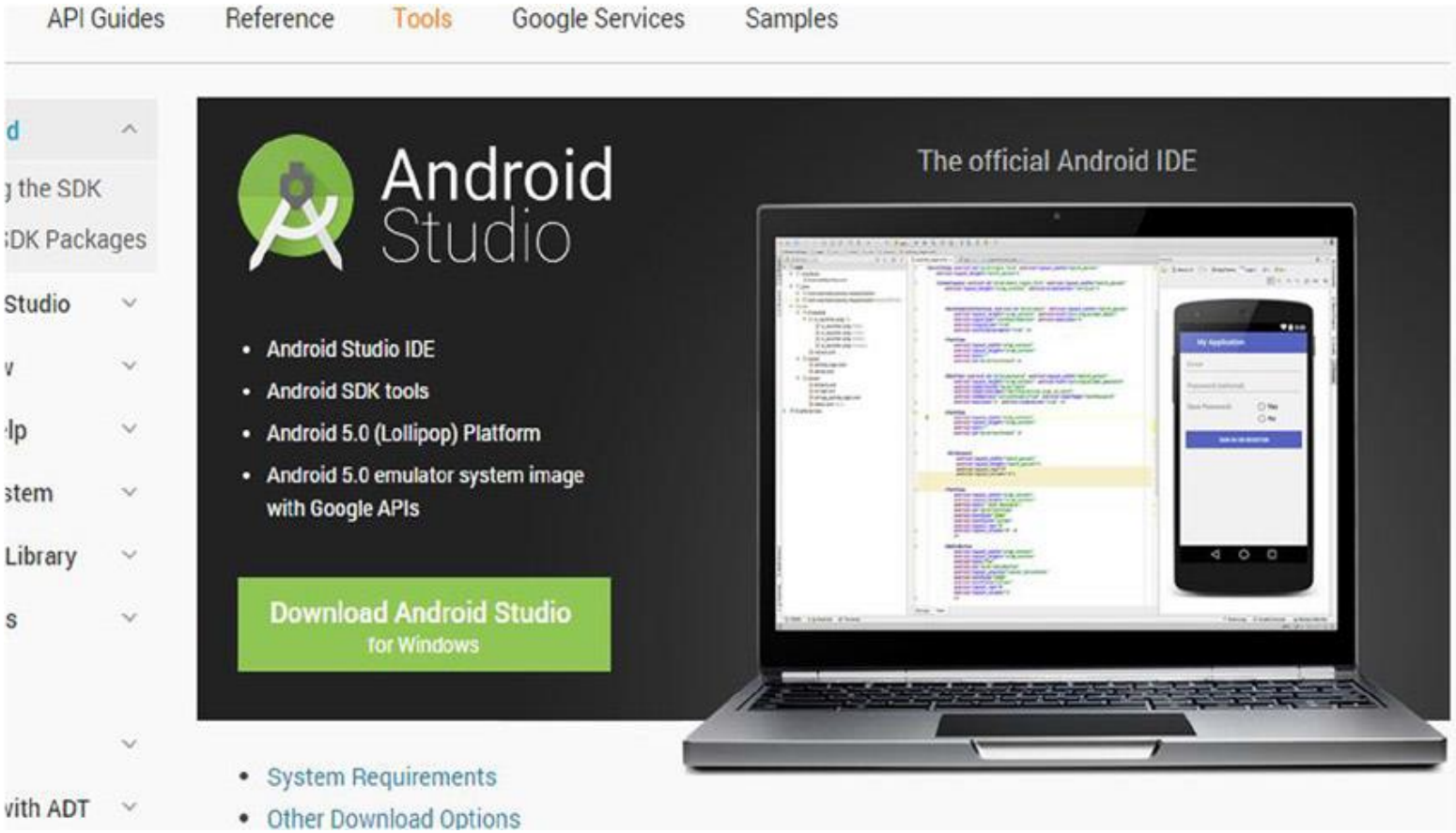



# Downloading and Installing

- In general Android Studio is at URL: http://developer.android.com/sdk/index.html
- The package downloaded includes the following: Android Studio bundle of IntelliJ IDEA, Built-in Android SDK, All related Android build tools and Android Virtual Device (AVD)



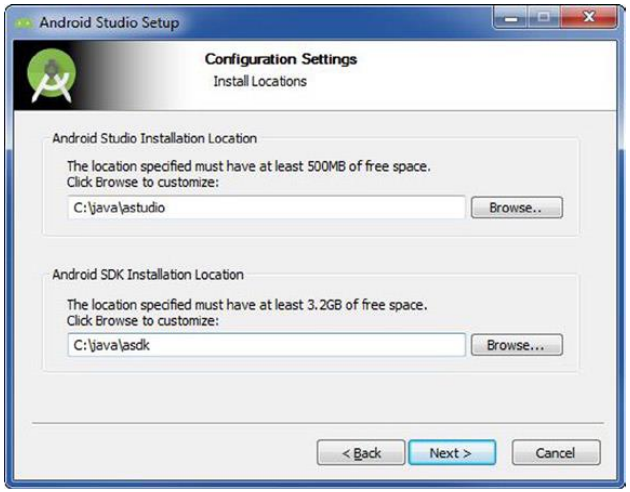
# Downloading and Installing

After download:

-for a Windows installation run the executable and follow the prompts to choose an installation path, and all the instalation options;

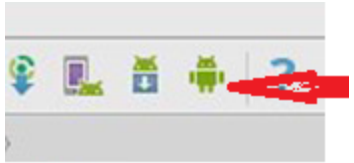
-for OS X, open the .dmg file and copy the Android Studio entry to your Applications folder.

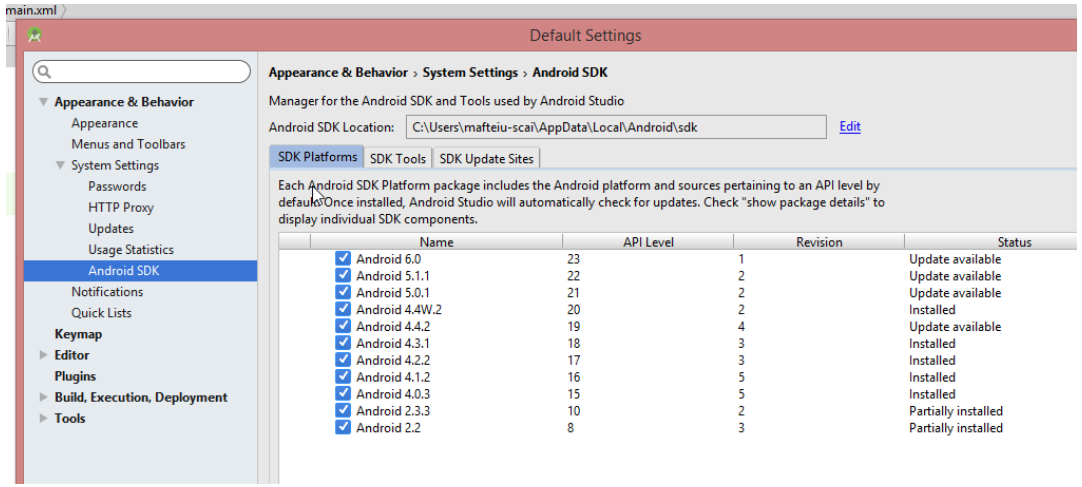
-under Linux, extract the contents of the .tgz file to your desired location.


# Downloading and Installing

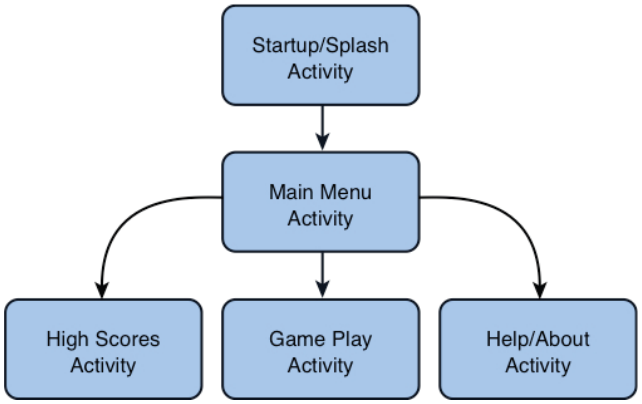
- **Downloading the Android SDK**: to build applications for Android, you need the Android SDK. As stated before, the SDK comes with the base tools. But, you can then, download the package parts that you need and/or want to use. For this you must use *Android SDK Manager.*





# Android's Fundamental Components

- **View**: user interface (UI) elements ( buttons, labels, text fields, etc) that form the basic building blocks of a user interface. *Everything you see is a view.*
- **Activity**: is a UI concept that usually represents a single screen in your application. It generally contains one or more views.
- **Fragment**: in the case of a large screen, it is difficult to manage all of its functionality in a single activity. *Fragments* are like sub-activities, and an activity can display one or more fragments on the screen at the same time.




*A simple game with five activities.*

- **Intent**: defines an "intention" to do some work. Examples of their use could be: start a service, launch an activity, display a web page, dial a phone number or answer a phone call. Intents are not always initiated by your application, they're also used by the system to notify your application of specific events, such as the arrival of a text message.

# Android's Fundamental Components

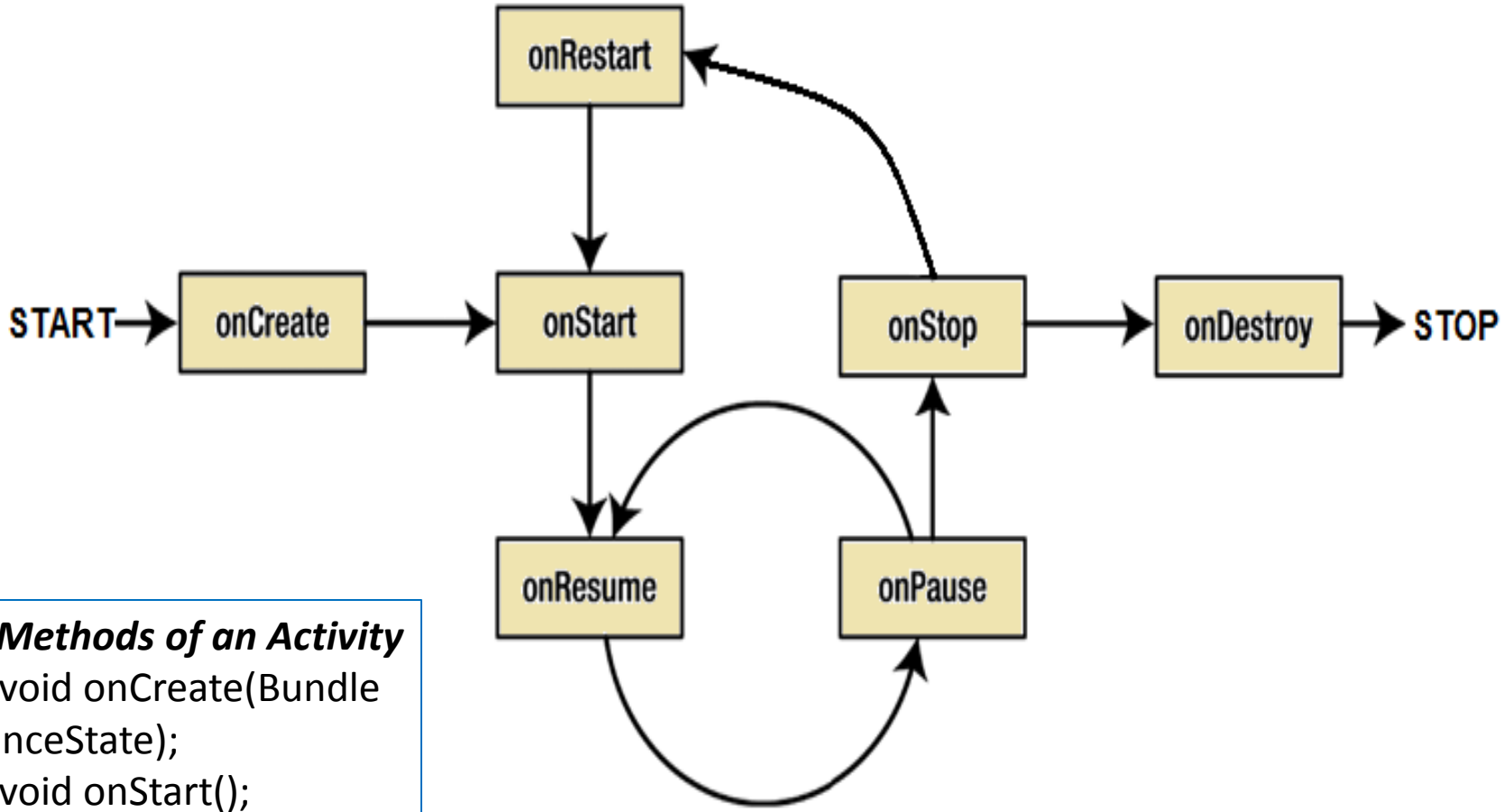
- **Content Provider**: due to data sharing among mobile applications on a device, Android defines a standard mechanism for applications to share data, such as a list of contacts.
- **Service**:

- local services: components that are only accessible by the application that is hosting the service.

- remote services: services that are meant to be accessed remotely by other applications running on the device.

- **AndroidManifest.xml**: defines the contents and behavior of app. For example, it lists app's activities and services, along with the permissions and features the application needs to run.
- **AVD *(Android Virtual Device)***: An AVD represents a device and its configuration. It allows developers to test their applications without hooking up an hardware Android device (smartphone or tablet). With AVD many different types of real devices can be emulated.

# Application Life Cycle




State transitions of an Activity

***Life-Cycle Methods of an Activity***

```
protected void onCreate(Bundle
savedInstanceState);
protected void onStart();
protected void onRestart();
protected void onResume();
protected void onPause();
protected void onStop();
protected void onDestroy();
```

# Simulation vs. Emulation

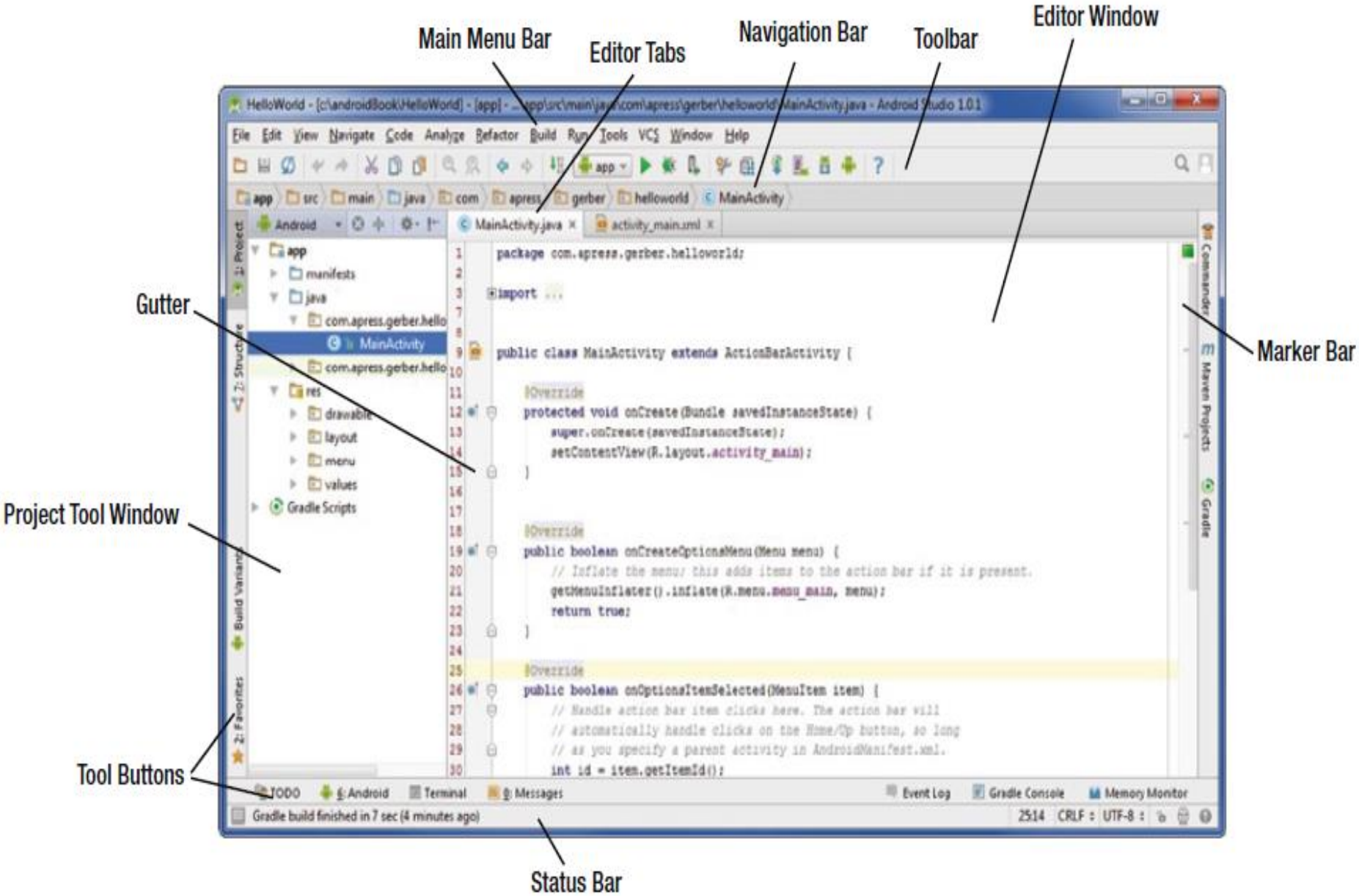
- **Simulation:** the simulated system behaves *similar to* real system. It provides the basic behaviour of a real system, but not all the rules of the real system.
- **Emulation:** the emulated system behaves *exactly like* real system, and respect all the rules of the real system. It is effectively a complete replication of a real system that operate in a different environment

---

- *Simulation: Corona SDK ⇔ Emulation: AVD Android Studio*

- *In Android Studio: AVD Manager is a utility provided by Google which allows to create emulated Android devices. Thus, can be created as many devices are desired, with different hardware specifications and with different screen resolutions.*

# Navigating Android Studio




*Android Studio's integrated development environment*

- Code Folding: hide particular blocks of code

```
3   import android.app.Activity;
4   import android.support.v7.app.AppCompatActivity;
5   import android.os.Bundle;
6   import android.view.Menu;
7   import android.view.MenuItem;
8   import android.widget.EditText;
9   import android.widget.Button;
10  import android.view.View;
11
12  public class MainActivity extends Activity
```

```
31
32      private void setupButtons() {...}
42
43      @Override
44      public void onClick(View v) {...}
65
66      private double plus(double n1, double n2) { return n1 + n2; }
```

*folding outline*

```
2
3   import ...
11
```

```
31
32      private void setupButtons() {
33          Button b = (Button) this.findViewById(R.id.plusButton);
34          b.setOnClickListener(this);
35          b = (Button) this.findViewById(R.id.minusButton);
36          b.setOnClickListener(this);
37          b = (Button) this.findViewById(R.id.multiplyButton);
38          b.setOnClickListener(this);
39          b = (Button) this.findViewById(R.id.divideButton);
40          b.setOnClickListener(this);
41      }
42
43      @Override
44      public void onClick(View v) {...}
65
```

- Code Completion

```
6   public class Sandbox {
7
8       private Li
9   }
10
```

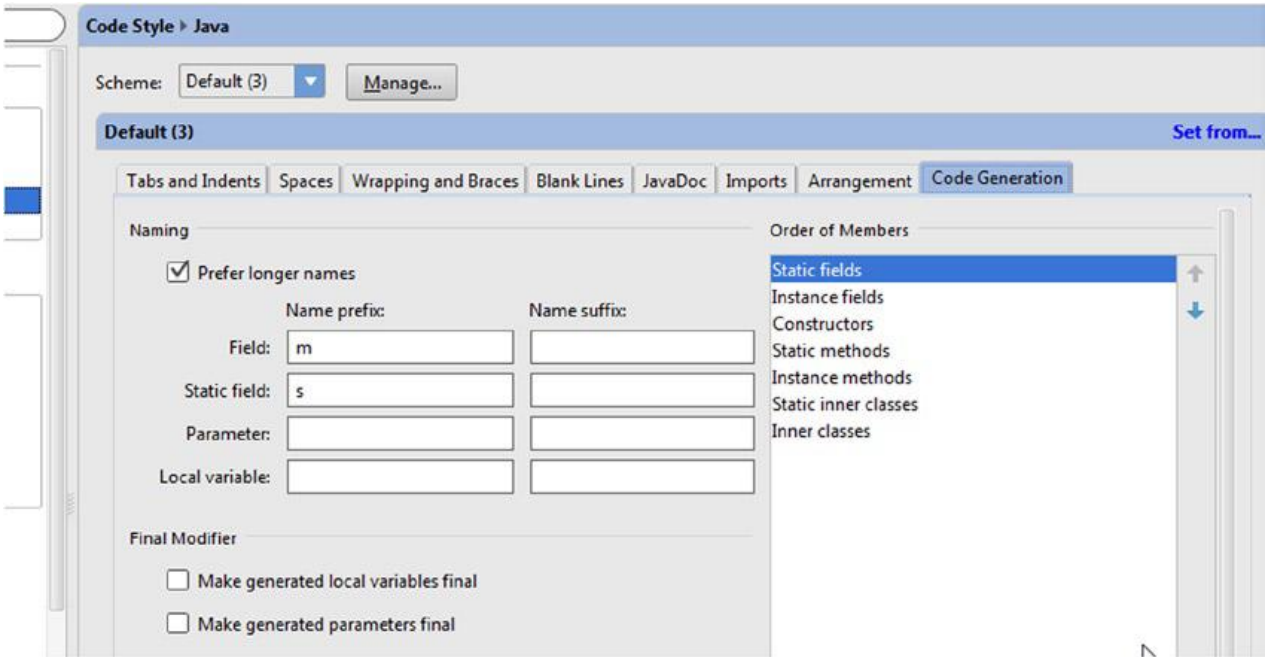
```
LinkedHashMap<K, V> (java.util)
LinkedHashSet<E> (java.util)
LinkedList<E> (java.util)
List<E> (java.util)
ListIterator<E> (java.util)
ListResourceBundle (java.util)
LinkageError (java.lang)
LinkedBlockingDeque<E> (java.util.concurrent)
LinkedBlockingQueue<E> (java.util.concurrent)
LineNumberReader (java.io)
Press Ctrl+Period to choose the selected (or first) suggestion and insert a dot afterwards >>
```

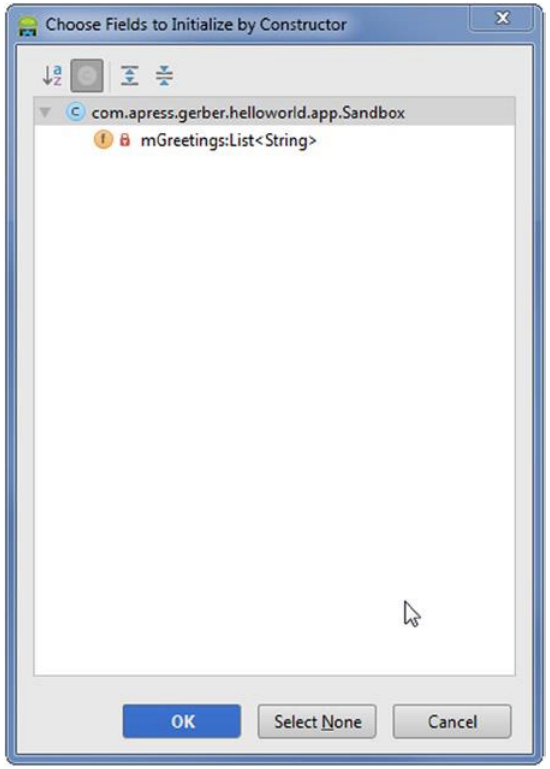
```
8   public class Sandbox {
9
10      private List<Str
11  }
12
```

```
String (java.lang)
StringTokenizer (java.util)
StrictMath (java.lang)
StringBuffer (java.lang)
StringBuilder (java.lang)
StringIndexOutOfBoundsException (java.lang)
StreamHandler (java.util.logging)
StreamCorruptedException (java.io)
StreamTokenizer (java.io)
StringCharacterIterator (java.text)
```

```
Documentation for String
< Android API 19 Platform >
public final class
String
extends Object
```

- **Code Generation:** generate methods (constructors, getters, setters, equals(), hashCode(), toString(), and so on).




**Styling The Code**:

Auto-Indent Lines

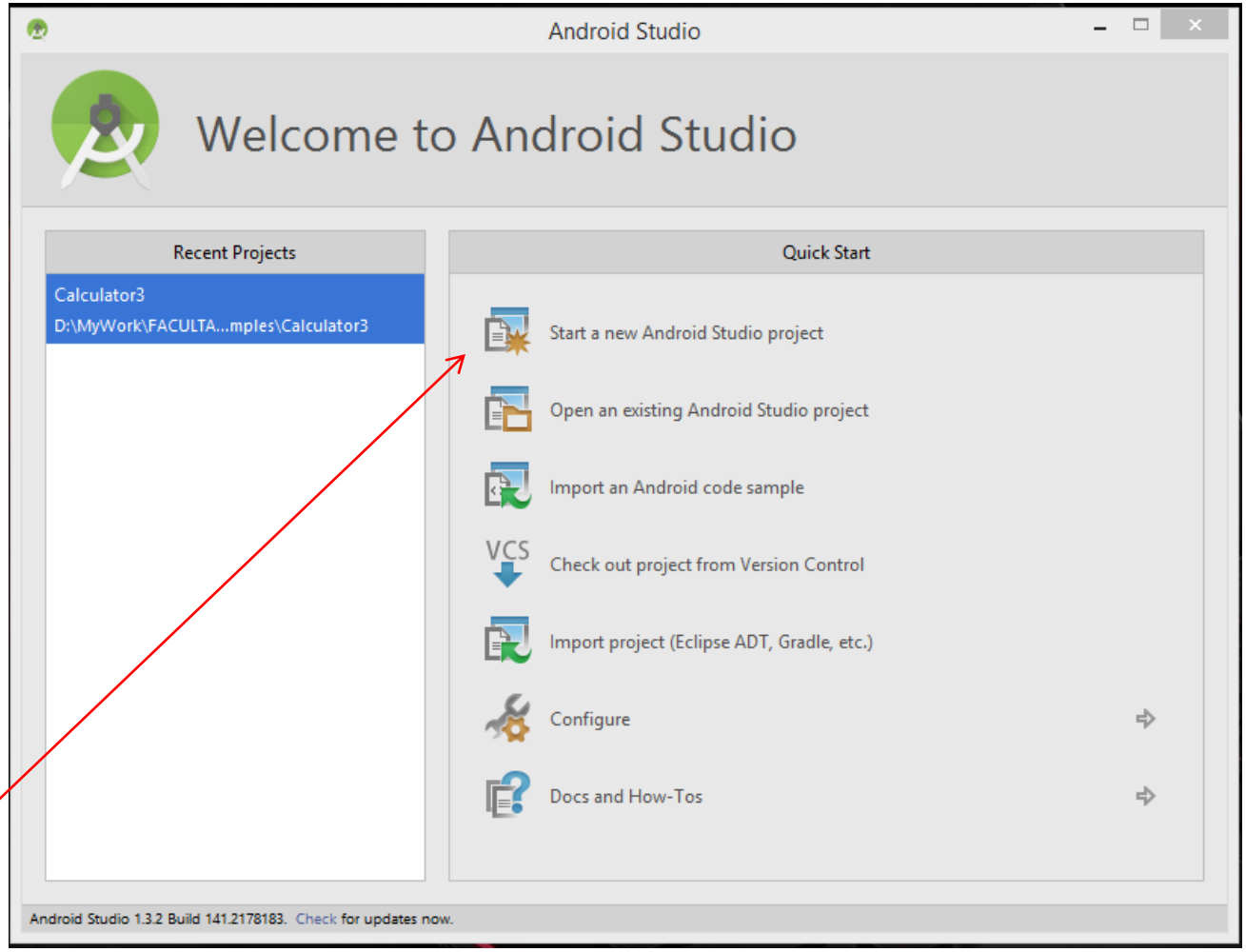
Rearrange Code

....

# Android Studio

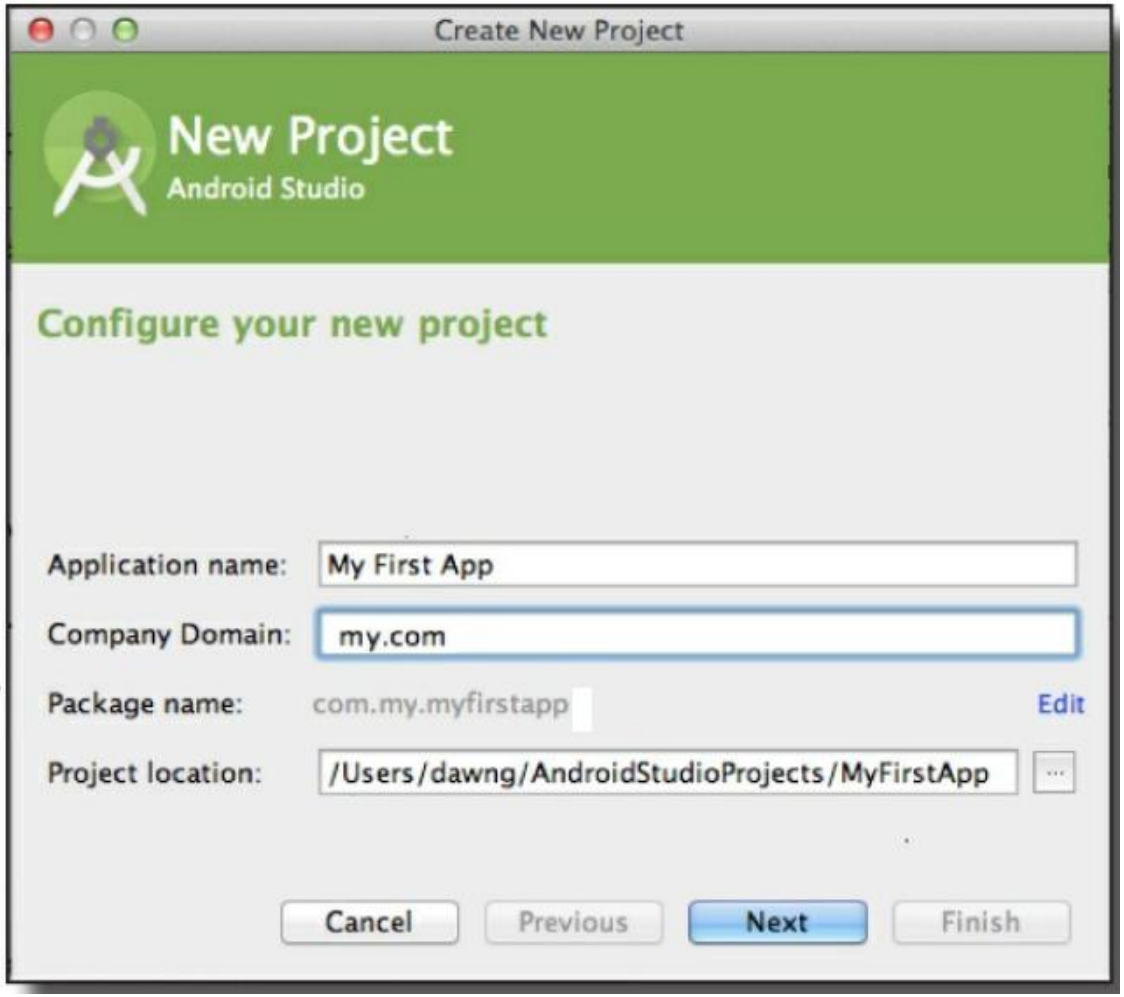
# *Build a First Basic App*

# 1. Open Android Studio



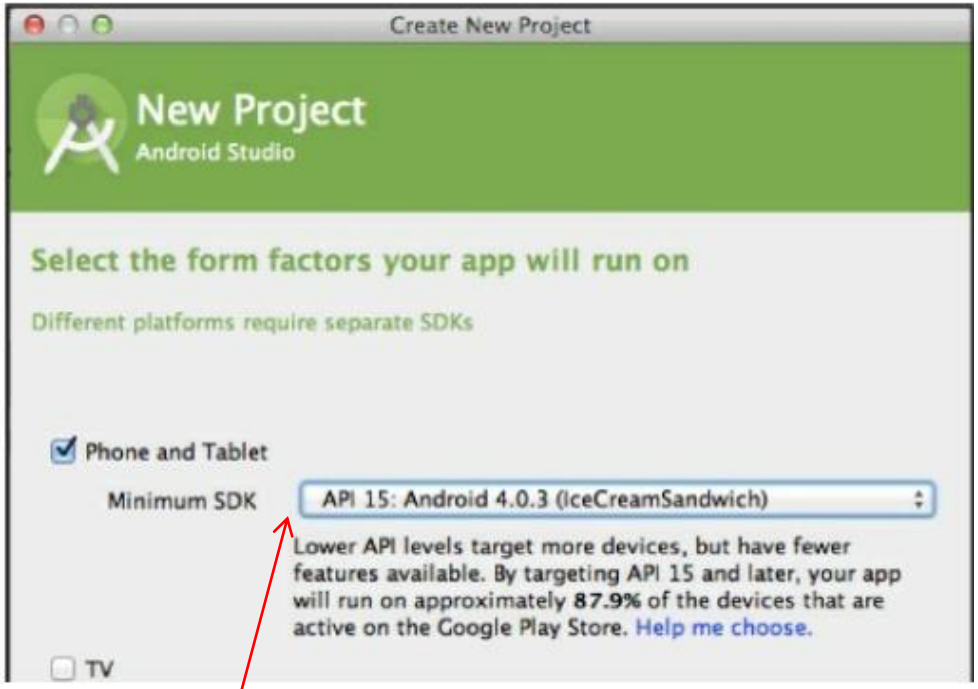

# 2. Start a new Android Studio Project

3. Enter an application name of “My First App”, a company name of “my.com”, and accept/change the default project location.




package name = combination(comp.domain, app name)

# 4. Specify the API level

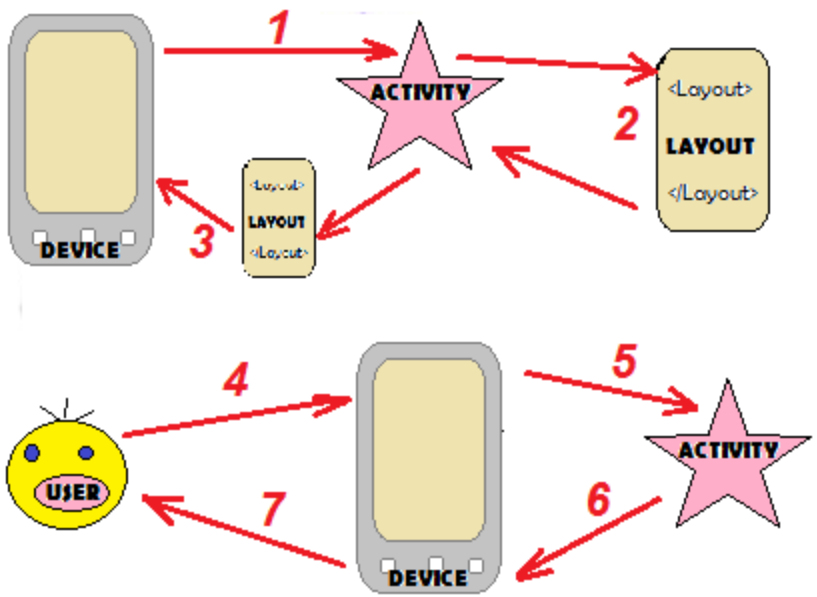



*The app will run only on devices with specified API level or higher*

| Version | Codename | API level |
|---|---|---|
| 1.0 | | 1 |
| 1.1 | | 2 |
| 1.5 | Cupcake | 3 |
| 1.6 | Donut | 4 |
| 2.0 | Eclair | 5 |
| 2.01 | Eclair | 6 |
| 2.1 | Eclair | 7 |
| 2.2.x | Froyo | 8 |
| 2.3 - 2.3.2 | Gingerbread | 9 |
| 2.3.2 - 2.3.7 | Gingerbread | 10 |
| 3.0 | Honeycomb | 11 |
| 3.1 | Honeycomb | 12 |
| 3.2 | Honeycomb | 13 |
| 4.0 - 4.0.2 | Ice Cream Sandwich | 14 |
| 4.0.3-4.0.4 | Ice Cream Sandwich | 15 |
| 4.1 | Jelly Bean | 16 |
| 4.2 | Jelly Bean | 17 |
| 4.3 | Jelly Bean | 18 |
| 4.4 | KitKat | 19 |
| 4.4 | KitKat (with wearable extensions) | 20 |
| 5.0 | Lollipop | 21 |

## 5. Specify Layouts and Activities

- Layouts define how the user interface is presented.
- Activities define actions.




**1. The device launches your app and creates an activity object.**

**2. The activity object specifies a layout.**

**3. The activity tells Android to display the layout on screen.**

**4. The user interacts with the layout that's displayed on the device.**

**5. The activity responds to these interactions by running application code.**

**6. The activity updates the display...**

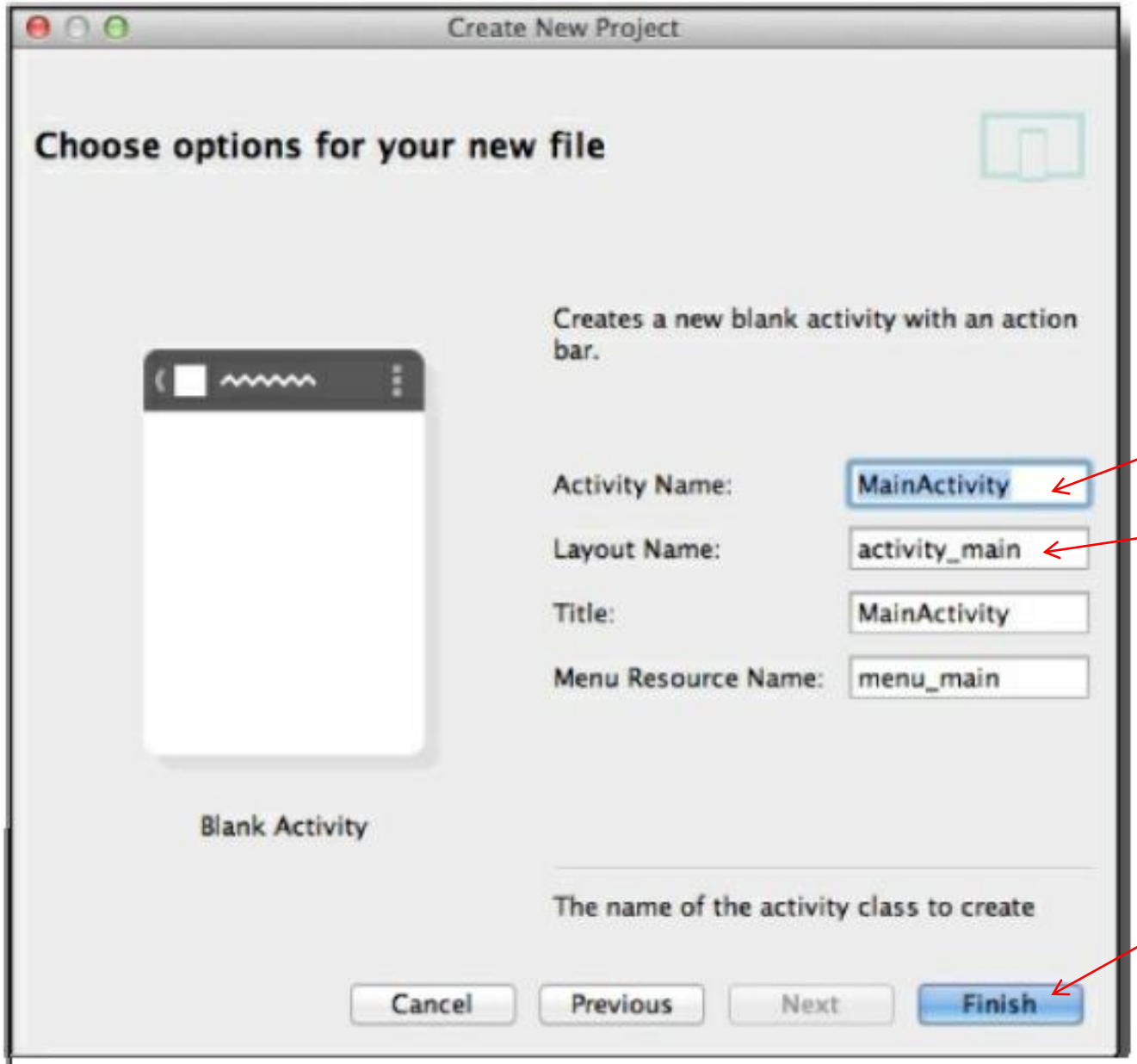
**7. ...which the user sees on the device.**

## 5.1 Create an Activity

create an app with a basic activity


*Select Blank Activity option*

## 5.2 Configure the Activity

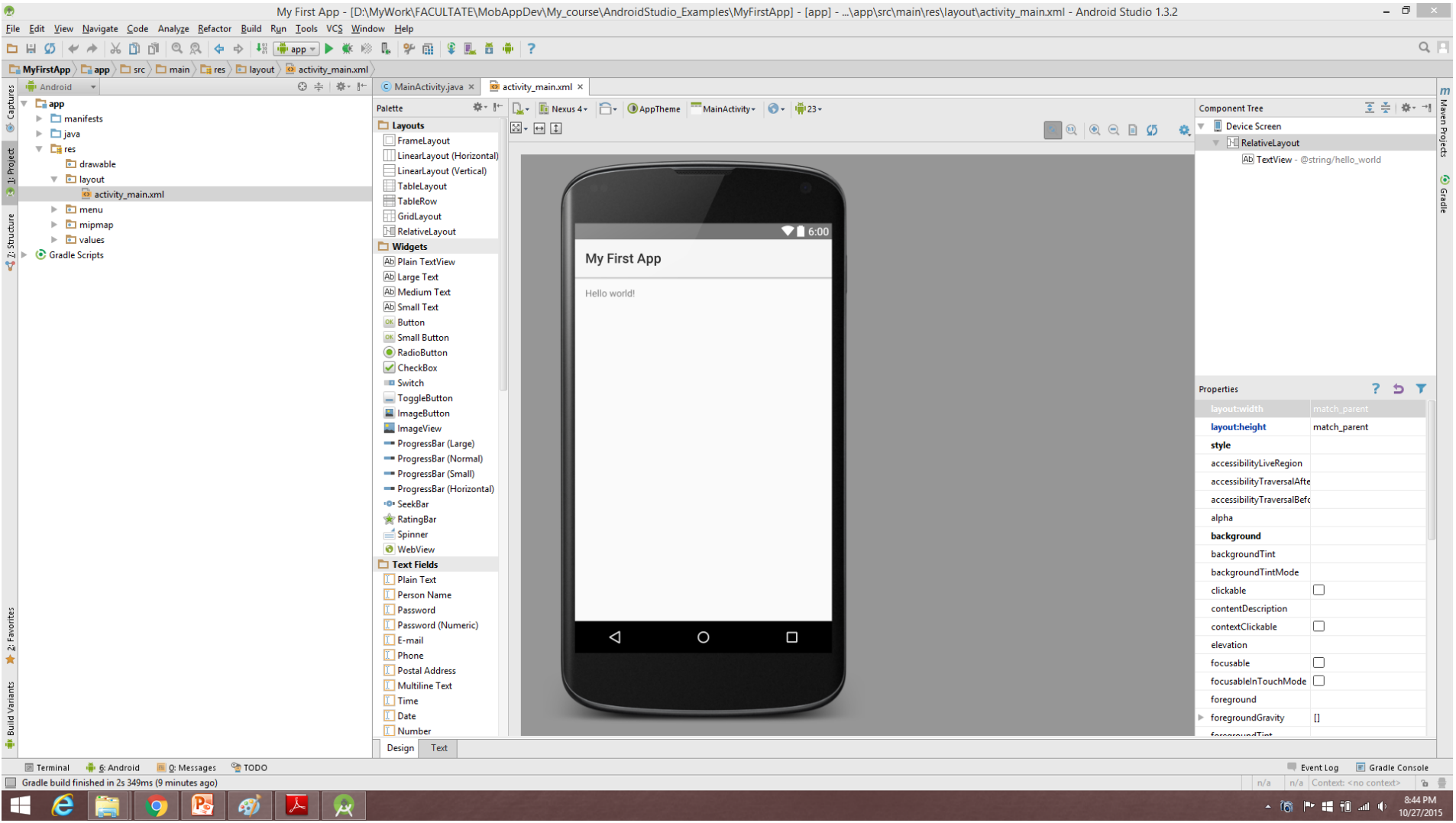



*MainActivity.java* class file

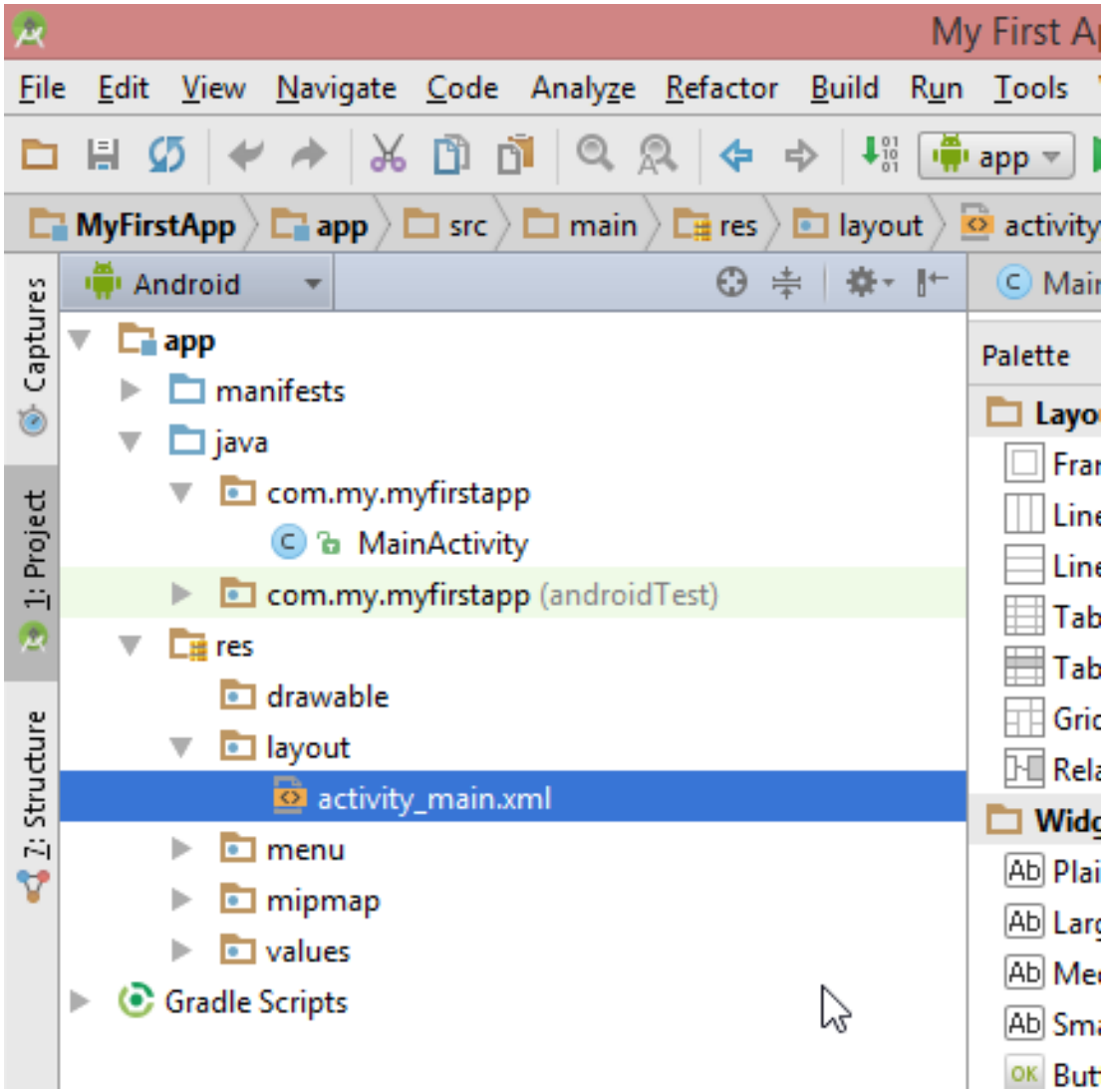
*activity_main.xml XML file*

*Android Studio will build the app.*

Review: **The Android Studio wizard created a project for MyFirstApp, configured to some specifications. It created a basic activity and layout with template code, with sample "Hello world!" text in the layout.**



**Android Studio creates a complete folder structure for MyFirstApp Project that includes different types of files (with collapse and expand function)**



**Android Studio creates a complete folder structure for MyFirstApp. For example:**

**-the *root* folder has the same name as project;**

**-the *build* folder contains files that Android Studio creates for you. You don't usually edit anything in this folder;**

**-the *source* folder contains source code you write and edit;**

**-every Android project needs a file called *R.java*, which is created for you and it lives in the generated folders. Android uses it to help it keep track of resources in the app;**

**-the *java* folder contains any Java code you write. Any activities you create live here;**

**-the *res* folder contains system resources. The *layout* folder contains layouts and the *values* folder contains resource files for values such as strings;**

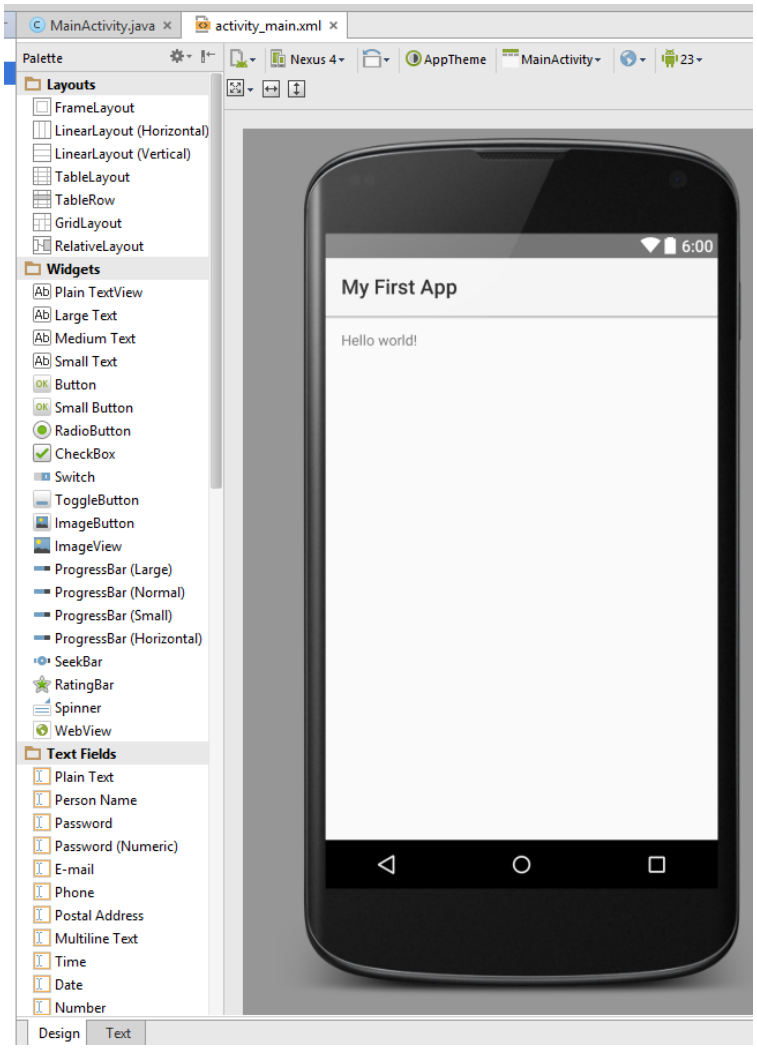
**-*MainActivity.java* defines an activity that tells Android how the app should interact with the user;**

**-*activity_main.xml* defines a layout that tells Android how the app should look;**

**-every app must include *AndroidManifest.xml* file that contains essential informations about the app (what components it contains, required library and other declarations);**

**-*string.xml* file contains strings such as app name and any defaults text values;**

# An important feature for layouts editing: Code Editor vs Design Editor

# Understanding *activity_main.xml*

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent“
        android:layout_height="match_parent“
         android:paddingLeft="@dimen/activity_horizontal_margin“
         android:paddingRight="@dimen/activity_horizontal_margin"
         android:paddingTop="@dimen/activity_vertical_margin"
         android:paddingBottom="@dimen/activity_vertical_margin"
        tools:context=".MainActivity">

  <TextView
    android:text="@string/hello_world"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>

</RelativeLayout>
```

Make the layout the same width and height as the screen size on the device.

Add padding to the screen margins.

Include a `TextView` GUI component for displaying text.

Display the text value of a string resource called `hello_world`.

Make the text wrap horizontally and vertically.

## Understanding *MainActivity.java*

```
package com.my.myfirstapp;

import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }
}
```

package name.

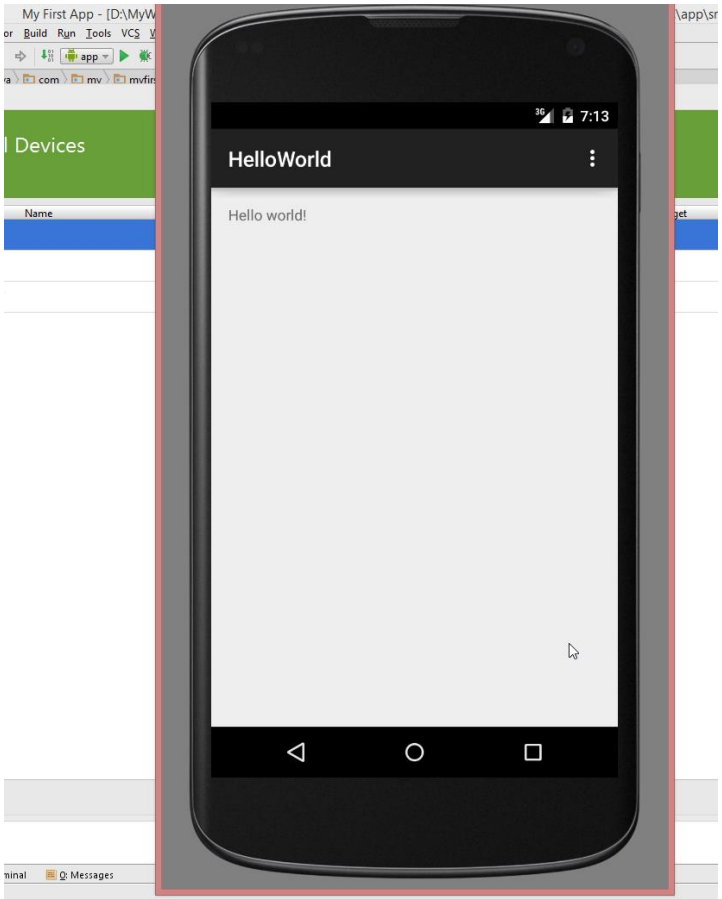
Android classes used in `MainActivity`.

`MainActivity` extends the Android class `android.app.Activity`.

Implement the `onCreate()` method from the `Activity` class. This method is called when the activity is first created.

Specifies which layout to use.

***Run MyFirstApp in the Android emulator :*** allows to run app on an Android virtual device (AVD). The AVD behaves just like a physical Android device. Thus, can be set up numerous AVDs, each emulating a different type of device.




**The emulator is an application.**

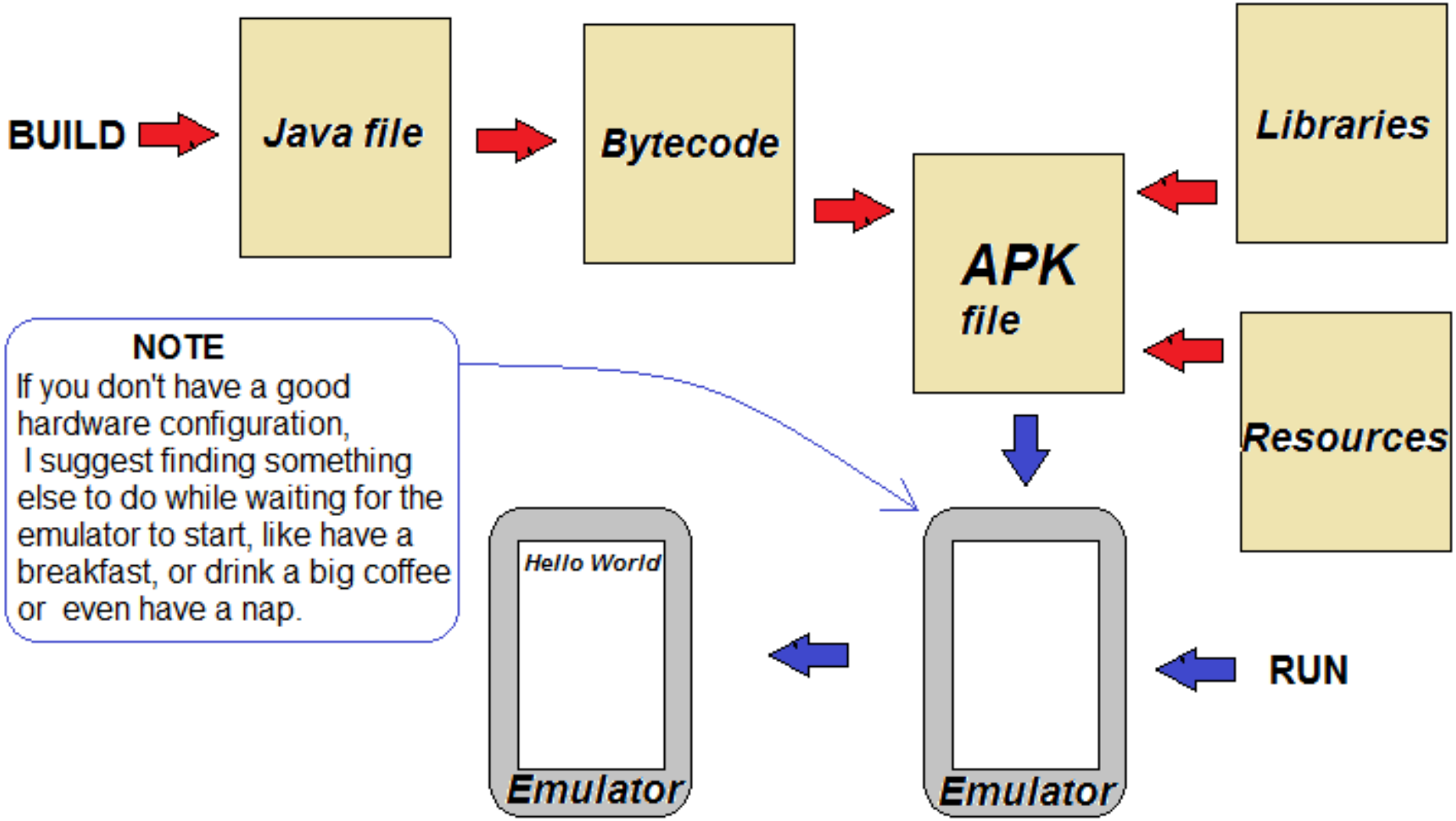
It looks just like a phone running on your computer.

Thus, it recreates the exact hardware environment of an Android device (CPU, memory, sound chips, video display etc)

It is built on an existing emulator called QEMU (similar to other virtual machine like VirtualBox).

# ***Understanding the Android emulator*** *: Compile, Package, Deploy, Run*

An APK file is an Android application package. It's basically a JAR or ZIP file for Android app.



**? Why the Android Emulator is so slooooooow ?** An Android Emulator uses an open source application called QEMU (or Quick Emulator) to emulate the entire Android hardware device, so it has to do a lot of work for each operation in part.

! On the other side, the iPhone Simulator runs much faster because all of the code for iOS is compiled to run natively on the Mac and the iPhone Simulator runs at Mac-native speed.

**? How to speed up Android development ?**

**1. Use a real device:** *"Developer options" and check the "Stay Awake" option.*

**2. Use an emulator snapshot:** *Tools→Android→AVD Manager→Edit AVD and check the "Store a snapshot for faster startup" option.* (improve booting Emulator using a copy of memory)

**3. Use Host GPU:** *Tools→Android→AVD Manager→Edit AVD and check the "Use host GPU" option.* (by using PC's graphic card for OpenGL)

**4. Use hardware acceleration:** On(ly) Intel x86 CPU machine, the Emulator can run the Android machine code instructions directly on Intel CPU using HAXM (Intel's Hardware Accelerated Execution Manager) HAXM is a hypervisor ⇔ switch CPU into a special mode to run virtual machine instructions directly

<u>Note: HAXM should be installed and will only run on Intel processors that support Intel Virtualization Technology</u>